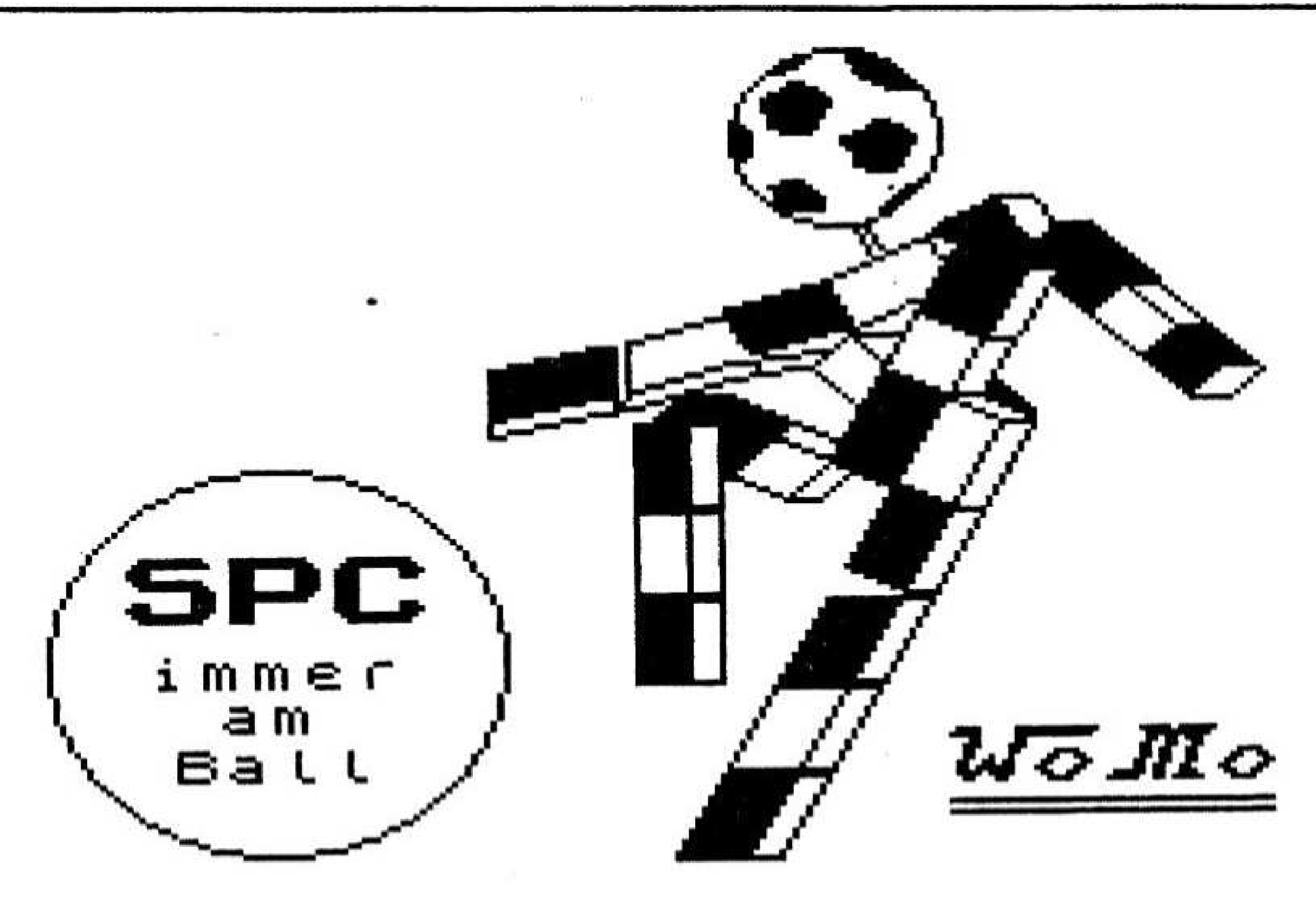
SPECTRUM PROFI CLUB

Rainbow User



Inhalt:

Wolfgang Haller
Ernastr. 33
5000 Köln 80
Tel. 0221/685946

INFO
7/90

Smalltalk...

Mein Gott - war das ein Wirbel mit der letzten "Rainbow User". Die zahlreichen Anrufe haben uns gezeigt, daß eine wirklich große Nachfrage nach Informationen rund um den Spectrum besteht. Vielen Dank übrigens auch für die vielen Genesungswünsche (wir waren ja beide krank!), aber daran hat's nicht gelegen, daß die "RU" so spät bei euch ankam. Die "Rote Karte" geht hier eindeutig an die Post, deren Umstellungen im gesamten Kölner Raum zu Riesenverzögerungen geführt haben. Im Klartext: Drucksachen werden von nun ab gesondert von der "normalen" Post behandelt und auch nicht mehr maschinell sortiert. Leider gibt es keine Alternativen, die Post hat das Monopol auf den Briefverkehr. Also aufgepaßt: Wer jetzt heiraten möchte und seine Karten per Drucksache verschickt, dem kann es passieren, das die Hochzeitsgäste ausbleiben, weil sie die Post nicht früh genug erhalten haben (kein Witz, so geschehen in Köln, im Juni 1990). Wir hoffen aber, das auch dies sich wieder einpendeln wird.
Aufgefallen ist sicher auch, daß die letzte Ausgabe auf Recycling-Papier kopiert wurde, unser Beitrag zum Umweltschutz. Die etwas ungewöhnliche Heftung war aber nur eine Notlösung.
Positiv: schon wieder 16 Seiten. Und wer trotzdem seinen Artikel vermißt, den verweise ich jetzt schon auf die nächste Ausgabe. Und: schickt weiter eure Artikel. Nur so bleibt die "RU" ein für alle interessantes Info.
Zum Thema "Sam Coupe" verweise ich heute auf den Artikel von Ian D. Spencer auf Seite 3. Ich kenne das "Gerät" und kann nur sagen, das es seinen Platz verdient hätte. Einen besseren 8-Bitter kenne ich zur Zeit nicht.
Dennoch hat der Spectrum bei uns immer noch seinen festen Platz, nicht zuletzt, weil er einfach "grandios" ist.
Noch ein Dankeschön an einige von euch für die Grüße im Computer-Flohmarkt.

Neuerscheinungen

Die Fußball-WM hat auch ihre Spuren bei den Programmierern hinterlassen. Gleich sieben (!!) Neuerscheinungen widmen sich dem Thema Fußball: Adidas Championchip Football (Ocean), Five A Side Football (Silverbird), Italy 1990 (US Gold), Kenny Dalglish Soccer Match (Impressions), Manchester United (Krisalis), World Cup Soccer '90 (Virgin) und World Cup Year '90 (Empire). Doch es gibt auch noch andere: Blinkey's Scarey School (Zeppelin), Cyberball (Domark), Defenders Of The Earth (Enigma), Escape From The Planet Of The Robot Monsters (Price), Grand Slam Tennis (Grandslam), Heavy Metal (Access), Hong Kong Phooey (HiTec Software), Judge Dredd (Virgin), Live And Let Die (Encore), Midnight Resistance (Ocean), Monthy Phyton's Flying Circus (Virgin), Raster Runner (Mastertronic), Ruff & Reddy (HiTec Software), Satan (Dinamic), Stormlord II (Hewson), The Cycles (Accolade) und Vendetta (System 3).
Auf für den SAM sind einige Programme in Arbeit: 3D Snooker (Players), Defenders Of The Earth (Enigma, erste Abbildungen in Crash 7/90, Seite 25!!), Fun School II (Database), Granny's Garden (Rickitt Educational Media), Kick Off (Anco), Play It Again SAM (MGT), Puncman (Rickitt Educational Media), Snowball In Hell (Atlantis), Spanish Gold (Rickitt Educational Media), Superleague (Players), The Race (Players) und World Boxing Manager (Goliath).

Tips und Tricks...

Dirk Mayer aus 5000 Köln 71, Lillerstr. 2, hat wieder ein paar Pokes zu Spielen der Crash-Cassetten herausgefunden: Doomskull (46758,0 = unendliche Leben) und Kemshu (25790,0 = unendlich Zeit).
Von Patrick Thiel aus 4796 Salzkotten, Königsberger Str. 11 kommt der Poke zu Target Renegade (59964,201 = unendliche Leben). Weiteres von Patrick auf den Seiten 12 und 13.
Noch ein paar Cheats: Mission Fallout (im Titelscreen Break drücken, es erscheint ein Speed-Menue. Speed wählen und Feuertaste. Es folgt ein Hardness-Menue. Härte wählen und Feuertaste, dann spielen.). Shinobi (Tasten neudefinieren in: G,R,U,T und S = unendliche Leben).

Diesmal fange ich mit einer schlechten Nachricht an. Während ich diesen Beitrag schrieb (Mitte Juni), wurde gerade angekündigt, das MGT (der Hersteller vom SAM) Konkurs angemeldet hat. Zur Zeit sucht ein Gerichtsbeauftragter nach zahlungskräftigen Firmen, die MGT übernehmen können. Wenn dies gelingt, hat der SAM eine sichere Zukunft, wenn nicht, dann ist alles sehr ungewiß. Schade, das dieses 'Cashflow'-Problem gerade jetzt zuschlägt, wo SAM in verschiedenen Ladenketten Fuß gefaßt hat und die Zukunft sehr gut aussah. Mehr als die Daumen drücken können wir nicht tun.
Jetzt etwas praktisches für alle, die ihren Spectrum und SAM mit dem Kabel von meinem letzten Beitrag verbunden haben.
Hierbei handelt es sich um zwei kleine Programme, die es erlauben, 'Sreens' vom Specci auf den SAM zu übertragen und auf SAM Diskette abzuspeichern. Diese können dann natürlich mit dem 'FLASH'-Grafikpaket bearbeitet werden. Im Prinzip liest das Spectrum Programm die Spectrum RAM-Adressen von 16384 bis 23295 (wo der Bildschirm gespeichert ist) und überträgt die Bytes auf den SAM. Es gibt aber ein paar Besonderheiten. Eine ist die Routine ab Zeilennummer 1000 und beim SAM die Procedure 'sersync'. Diese Routine schickt erst 10mal den Wert 170 (10101010 in binär) und dann den Wert 85 (11001100). Dies bedeutet, das der SAM erst ein Bild akzeptiert, wenn er mehrere 170-Werte und einen 85-Wert empfangen hat. Wenn wir diese Routine nicht haben und zufällig ein paar Bytes vorher in der seriellen Schnittstelle des SAM sitzen (es können bis 3 sein), dann werden diese 'Schrott-Bytes' den Filenamen verfälschen und das Bild am Schirm verschieben. Das bringt uns zur zweiten Besonderheit. Der Filename vom Specci wird nach den sync bytes automatisch zum SAM übertragen, dies wird vom SAM in den Zeilen 1030 bis 1060 gelesen. Die Werte für "b" in Zeile 1070 sind für einen 512K SAM, für einen 256K SAM werden die Werte 245760 bis 252671 benutzt.
Einen Bildschirm mittels eines Basic-Programms zu übertragen, ist sehr langsam, dafür sieht man aber, wie alles funktioniert. Viel Spaß und Tschüß bis zum nächstenmal.

Ian D. Spencer, Fichtenweg 10c, 5203 Much, Tel. 02245/1657

SPECTRUM

```
  10 REM SPECTRUM/SAM SCREEN$
  20 DIM S$(10)
  30 OPEN #7;"b"
  40 INPUT "Screen Name ? ";S$
  50 LOAD *"m";1;S$
  60 GOSUB 1000
  70  FOR a=16384 TO 23295
  80  LET b=PEEK a
  90  PRINT #7;CHR$ b
 100 NEXT a
 110 CLOSE #7
 120 STOP
1000 FOR a=1 TO 10
1010  PRINT #7;CHR$ 170;
1020 NEXT a
1030 PRINT #7;CHR$ 85;
1040 FOR a=1 TO 10
1050  PRINT #7;S$(a);
1060 NEXT a
1070 RETURN
```

SAM

```
  10 REM * SPECTRUM SCREEN$ *
  20 MODE 1
  30 PRINT AT 0,7;"* SPECTRUM SCREEN *"
  40 DIM s$(10)
  50 getscreen
  60 SAVE S$ SCREEN$
  70 PAUSE 0
  80 STOP
1000 DEF PROC getscreen
1010  OPEN #7;"b"
1020  sersync
1030  FOR b=1 TO 10
1040   getchar
1050   LET S$(b)=cha$
1060  NEXT b
1070  FOR b=507904 TO 514815
1080   getchar
1090   POKE b,char
1100  NEXT b
1110 END PROC
2000 DEF PROC sersync
2010  LET a=0
2020  getchar
2030  IF char<>170 THEN GOTO 2010
2040  LET a=a+1
2050  IF a<4 THEN GOTO 2020
2060  getchar
2070  IF char<>85 THEN GOTO 2030
2080 END PROC
3000 DEF PROC getchar
3010  LET cha$=INKEY$#7
3020  POKE 60000,cha$
3030  LET char=PEEK 60000
3040 END PROC
```

Spieltip: Jack the Nipper 2

Zu dem in der Rainbow-User 2/90 von Harald R. Lack vorgestellten Gremlin-Spiel hat Paul Webranitz einiges herausgefunden.

Hallo Freak's

Heute was über ein Game, welches schon seit geraumer Zeit bei mir rumliegt, weil es ohne Superpoke's und Anleitung nicht spielbar ist.Es ist ein Suchfindundbring esirgendwohindannpassiertwasgame Namens " JACK THE NIPPER 2 ".

War bei Nipper 1 noch die logische Zuordnung der Teile halbwegs möglich, ist bei Nipper 2 das Erkennen der Teile schon Glückssache !

Das ganze spielt sich in 190 ! Bildern auf 4 Ebenen ab.Jede Menge Monster,welche dem armen Jack an den Kragen wollen.Jede Menge Schwierigkeiten,um von einem Bild in's andere zu gelangen. Hier hilft nur der Superpoke für ewiges Leben: 43251,0. Aber der allein genügt nicht. Mit 43227,250 und 34426,0 wird man unverletzlich. Aber auch dies genügt noch nicht. Ab und zu findet man eine Cocusnuß. Mit dieser kann man 16 mal werfen. 38306,0 ermöglicht unendliche Würfe. Zum Unverletzlichkeitspoke noch ein Hinweis: dieser wird erst nach dem ersten Zusammenstoß mit einem Monster wirksam. Jack ändert dann ständig seine Farbe. Die Kalebassen in einigen Bildern bewirken ebenfalls eine gewisse Zeit Unverletzlichkeit, können also unbeachtet bleiben. Zumal beim Aufnehmen die Cocosnuß flöten ist. Das gleiche gilt für die Dynamitstangen oder was das sein soll. Von diesen kann man 2 finden. Ich habe aber noch nicht raus, wie die gezündet werden. Die eine wird mit Sicherheit gebraucht, um den Geheimgang in Raum 63 freizusprengen, damit man in's letzte Bild gelangt.

Geheimgänge: In den Felsen sind ab und zu Gesichter eingehauen. Dahinter verbergen sich Geheimgänge (manchmal). Auch in den Kronen der Bäume.

Zu den Teilen: Welches Teil man mit sich rumschleppt, steht in der Speicherzelle 34878. Auch in diese kann man poken, jedoch wird das entsprechende Symbol in der Fußleiste nicht angezeigt. Es ändert sich nur beim Tausch der Teile. Hier die Code-Nummern: 246 = Schalnge, 247 = Kürbis, 248 = Ananas, 249 = Maus, 250 = Dose mit Deckel, 251 = Schokolade, 252 = Kette (Seil?), 253 = Dose mit Aufschrift GREY. Wozu die Teile dienen, habe ich noch nicht entdeckt. Wer was weiß, den bitte ich um Mitteilung.

Das einzige, was ich bisher erreicht habe um das Naughtyometer zu bewegen, ist in Bild 58 der Bienenstock welcher an dem Ast des grünen Baumstumpfes hängt. Schießt man diesen mit der Cocusnuß ab, wird der Schwarm frei und folgt einem auch in's Bild 57. Und damit zu den Bildnummern:

Diese stehen in der Speicherzelle 34885.Pokt man hier eine andere Nummer (0-190) kommt man, wenn man das Bild am Rand verläßt in dem Bild N+1 oder N-1 oder N+32 oder N-32 raus. Dazu noch was: das ganze spielt sich in 4 Ebenen ab. Diese Ebenen sind jeweils 32 Bilder lang. Will ich also von Raum 64 in den darunterliegenden Raum, müß ich 32 addieren, also 96 einpoken. Damit das ganze nicht so einfach ist, muß man allerdings beachten, ob in dem Bild wo Jack rauskommen soll nicht eine Wand ist! Sonst hat man Jack eingemauert!!! Wie ihr seht, geht das ganze nicht ohne Multiface.

Die Schnuller, welche man findet, sind Zusatzleben. Mit unserem Poke brauchen wir die aber nicht. Wer sich nun also in das Spiel stürzen will, dem sei empfohlen, sich einen Plan der Räume anzulegen. Ansonsten verirrt er sich gar schröcklich. Und aus Bild 60 gibt es kein Entkommen, wenn man das richtige nicht dabei hat!! (Tempel mit Geheimgang rechts oben)

P.S.: Inzwischen habe ich doch noch was rausgeknobelt. Der Negergeist in Raum 44 darf nicht abgeschossen werden, sondern muß in's Wasser fallen. Die Brücke in

Raum 112 verschwindet, wenn man die Schlange dabei hat (die Schlange verschwindet auch). Mit der Maus jagt man in Raum 56 den Elefanten auf den Baum und in Raum 57 und der Dose (welche man beim Bienenstockabschießen dabeihaben sollte) die Neger aus der Hütte. Der Kürbis muß in Raum 68 zu den Katzen. Die Kette in Raum 167. Die Dose mit der Aufschrift GREY nach Raum 34. Das ganze muß aber in der richtigen Reihenfolge ablaufen, sonst funktionierts nicht. Wie diese aussehen muß, habe ich allerdings noch nicht raus.

Mit der Ananas und einer Stange Dynamit wird der rasende Wirbelwind in Raum 89 gekillt. Ist das Naughtyometer ganz voll, ist in Raum 60 der Geheimgang offen. Noch was: Aus unerfindlichem Grund wird im MC laufend die Speicherzelle 49738 abgefragt und auch auf Null gesetzt. Nirgendwo jedoch wird in dieses Zelle etwas eingetragen. Diese Abfrage jedoch (In Programmteil 35469) verhindert, daß mit den aufgenommenen Teilen was angefangen werden kann (Crackschutz???). Hier sind am besten die beiden RET Z zu entfernen. Also 35497,0 und 35500,0.
Dennoch viel Spaß wünscht Euch PANTHERPAULE.

Und dann erreichte uns noch folgender Nachtrag von Paule:

Heureka! Endlich geschafft. Jack the Nipper ist gelöst!

Damit was passiert, wenn man was dabei hat, muß DOWN + Feuer gedrückt werden. Also hier die Reihenfolge was benutzt werden muß und wo:

Die Dose mit Deckel darf erst als letztes benutzt werden!!! Sonst geht nichts mehr. Auch die Schlange, welche die Brücke verschwinden läßt, sollte erst als vorletztes eingesetzt werden, sonst versperrt man sich den Rückweg.
Kürbis in Raum 68 zu den 3 Miezen. Seil in Raum 167. Dort auf den einzelnen roten Ast springen und wenn der kleine Indianer ziemlich genau unter dem Ast steht, DOWN + Feuer. Muß mehrmals ausprobiert werden, ehe es funktioniert. Mit der Dose (Aufschrift GREY) nach Raum 34. Dort hängt ein Männeken an einem Seil und zappelt. Auf den Ast oberhalb stellen und DOWN + Feuer. Falls man unterwegs irgendwo in einer Höhle mit einem Seil an der Decke und darunter Wasser einen stehenden Negergeist entdeckt (Raum 44), diesen anspringen und ins Wasser plumpsen lassen.

Die Schokolade sollte man dabeihaben, wenn man auf dem Wasser in der zweituntersten Etage auf den schwimmenden Klötzen fährt. Steht man auf einem solchen, ebenfalls DOWN + Feuer. Die Ananas und eine Stange Dynamit nach Raum 89 mit dem Wirbelwind.

Hier an den rechten Bildrand an die Säule stellen und warten bis der Wirbelwind Nipper erreicht hat. DOWN + Feuer. Die Maus nach Raum 56 zu dem müden Elefanten, welcher am Baum lehnt. DOWN + Feuer. Nun mit der Schlange nach Raum 112 (Brücke).

In den Räumen unter der Brücke liegt noch eine Schlange. Scheint übrig zu sein. Konnte jedenfalls nichts damit anfangen. Hoffentlich habt ihr nun noch die Dose mit dem Deckel dabei. Nach Raum 58 und den Bienenkorb an dem grünen Baum abschießen. Einen Raum zurück in die Hütte. DOWN + Feuer. Muß mit wechselnder Position gegebenenfalls mehrmals wiederholt werden, bis die Indianer aus der Hütte kommen.

Nun sollte das Naughtyometer ganz voll sein und Nipper kann sich auf den Weg zu Raum 60 machen. Ist man in diesen Raum gelangt, ohne daß alles gelöst worden ist, hilft nur ein Neubeginn. Aus diesem Raum gibt es keinen Ausgang. Nur der offene Geheimgang zum Tempel, falls alles gelöst worden ist.

Paul Webranitz, Borgasse 16, 5561 Kinheim, Tel. 06532/2607

Ersatztransistortypen

Hallo,

immer wieder lese ich, daß Leute Ersatztypen für verschiedene Transistortypen suchen. Da habe ich einmal meine Datenlisten durchgewühlt und stelle hiermit vor was ich gefunden habe. Wer noch andere Typen beschreiben kann, kann dies dann vielleicht in der nächsten Ausgabe nachholen.

TYP		V	A	W	B	F	Bemerkung
BC 183	N	30	200m	300m	100..850	150M	
BC 184 C	N	30	200m	300m	450..900	150M	rauscharm IF 3,6,9 und MD 1
BC 213 C	P	30	200m	300m	350..600	200M	IF 10,11
BC 214	P	30	200m	300m	200..400	200M	IF 1,4,5
BC 327 C	P	45	800m	625m	250..630	100M	
BC 337 C	N	45	1	625m	250..630	100M	
BC 636	P	45	1	800m	40..250	50M	
ZTX 213	P	30	200M	500m	125..550	200M	ZX 5
ZTX 313	N	15	500m	300m	40..120	240k	Schalt
ZTX 450	N	45	1	1	100..300	150M	
ZTX 510	P	12	200m	300m	40..150	400M	Schalt
ZTX 551	P	60	2	1	50..150	150M	MD 2
ZTX 650	N	45	2	1	100..300	100M	ZX 4

N und P bedeutet NPN oder PNP. Diese Eigenschaft ist ein Muß bei Ersatztypen. Das Nichtbeachten ist vergleichbar damit, in ein Kofferradio die Batterien falsch herum einzulegen. V ist die Spannung, die maximal am Transistor anliegen darf. Bei Ersatztypen vorzugsweise höhere Werte nehmen. A ist der maximale Strom durch den Transistor und P die maximale Leistung, die im Transistor in Wärme umgesetzt wird. Die beiden letztgenannten Werte sollten keinesfalls unterschritten werden, da Sinclair die "Macke" hat, die Bauteile immer bis an die Grenzen ihrer Leistungsfähigkeit "auszulutschen". Deshalb ist es auch schwierig, für die ZTX 551 und 650 Ersatz anzubieten. Diese sind nämlich eine Spezialität von FERRANTI: Die SUPER E-LINE Typen. Aber meine Listen sind nicht die aktuellsten. Vielleicht hat ein anderer Hersteller ein eigenes Verfahren entwickelt, das ähnlich leistungsfähig ist und von dem leichter Bauteile zu kriegen sind. Allerdings gibt es inzwischen auch Händler, die FERRANTI führen. Also mal umschauen, denn Ersatztypen sind eben nur Ersatz. Auf Anfrage sende ich auch ZTX 652 als Austauschtyp für ZTX 650 von meinem Ortshändler zu. B ist die Verstärkung des Transistors, meist sind höhere Werte weniger kritisch.F ist die Transitfrequenz. Das hat nichts mit Grenzverkehr zu tun, sondern ist die Frequenz, bei der ein Signal ohne Verstärkung durch den Transistor "geht" (lat.: transire oder so ähnlich). Auch hier im Zweifel höhere Werte nehmen. Bemerkungen sind besondere Eigenschaften der Type oder die Verwendung bei Sinclair.
Zu meinem Microdrive Schaltplan habe ich noch ein paar Nachträge: Die Diode D2 geht über E1 an B5. Die Spannungsversorgung liegt an A4 und nicht an A5. Die oberen Anschlüsse vom Schreibschutzschalter SW und der LED sind vertauscht. Das heißt SW über F5 an D4 und LED über F3 an R3. C5 ist entgegen dem Nachtrag von Volker Henschel in ru 3/89 doch ein Kondensator und zwar 47nF und auch die anderen "Zenerdioden" sind Kondensatoren. Vielen Dank, daß überhaupt jemand auf den Plan reagiert hat. Wie aus einigen Anfragen der letzten Ausgaben hervorgeht, haben ihn einige gar nicht gesehen...
Ein Nachtrag zu Dieter Hucke in ru 2/90: Bit 4 vom Port 254 ist doch die Tonausgabe. Dieses Bit liefert einen höheren Pegel als Bit 3. Am Tonein/ausgang der ULA hängen die MIC und EAR Buchse über RC-Glieder, also mehr oder weniger direkt. Der Lautsprecher dagegen hat eine Diode in Serie geschal- tet, deren Schwellspannung erst überschritten werden muß, bevor ein Ton entsteht. Deshalb ist SAVE am Lautsprecher leiser als BEEP.

Aus der Hobbytronic habe ich ein ROTRONIC WAFADRIVE bekommen. Wer Informationen, Bedienungsanleitung, Schaltbild o.ä. hat, möge mir bitte Kopien schicken gegen Portoerstattung (Kontonummer angeben).
Ebenso würde ich mich über Anleitung und Systemdiskette zum Beta-Disk-IF V4.12 sehr freuen.
Sozusagen als Fortsetzung zum Microdrive-Schaltplan folgt heute der vom Interface 1. Dazu einige Anmerkungen:
Die Bauteil- und Anschlußbezeichnungen habe ich von der Platine übernommen. Die Bauteilseite des Microdrive-Verbinders heißt A, genau wie die Adressleitungen der CPU. Ich hoffe das bringt niemanden durcheinander. Den Steckverbinder zum Spectrum habe ich nicht mit eingezeichnet. Die Zusammenhänge ergeben sich ja aus den Bezeichnungen.

Nun die Bauteileliste:

Widerstände R

Nr.	Wert
1	1k
2	
3	1k
4	3k9
5	1k
6	3k9
7	47
8	3k9
9	10k
10	2k2
11	1k
12	12k
13	680
14	1k
15	12k
16	3k9
17	2k7
18	680
19	1k
20	2k7
21	10k
22	680
23	680
24	4k7
25	10k
26	1k
27	10k
28	6k8
29	6k8
30	270
31	1k
32	680
33	10k
34	330
35	47k
36	10k

Quarz X

Nr.	Wert
1	8MHz

"chips" IC

Nr.	Typ
1	ULA
2	ROM
3	74LS260
4	74HCU04

Kondensatoren C

Nr.	Wert
1	47 µ
2	47 µ
3	22p
4	100p
5	?p
6	47 µ
7	330p
8	330p

Spule L

Nr.	Wert
1	22 µ

Dioden D

Nr.	Typ
1	power SI
2	power SI
3	DUS
4	DUS
5	
6	ZPD4v3
7	ZPD4V3
8	ZPD4V3
9	DUS
10	DUS
11	DUS

Transistoren Q

Nr.	Typ
1	BC213C
2	ZTX510
3	BC184C
4	BC213
5	BC213
6	BC184
7	
8	
9	BC184
10	BC183
11	BC183

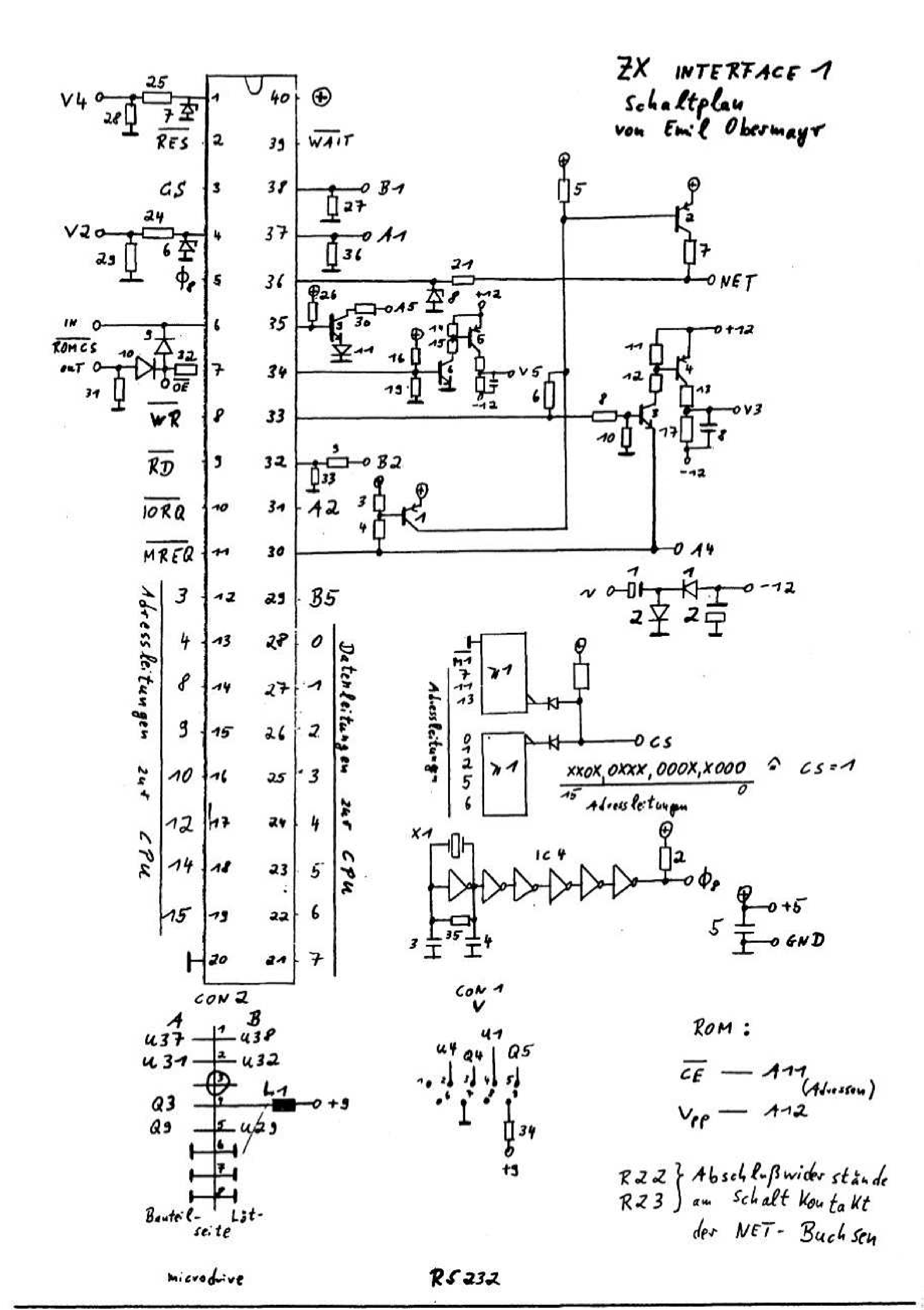
(frei verdrahtet)

Wie üblich schwanken die Bauteilwerte von Gerät zu Gerät und von einer Issue zur nächsten.

Da ich den Steckverbinder auf den Schaltplänen weggelassen habe und er in den neuen Handbüchern soweit ich mich recht erinnere nicht mehr abgebildet ist hier zusammen mit dem vom ZX81 aus einen Blick:

Bauteile A	81	Pin	81	81	Lötseite B
A15		1			A14
A13		2			A12
D7		3	1		+5V
NC	MEMRAM	4	2		+9V
		5	3		
D0		6	4		GND
D1		7	5		GND
D2		8	6		CLK
D6		9	7		A0
D5		10	8		A1
D3		11	9		A2
D4		12	10		A3
INT		13	11	A15	IO ULA
NMI		14	12	A14	GND
HALT		15	13	A13	FBAS
MEMREQ		16	14	A12	Y
IOREQ		17	15	A11	V
RD		18	16	A10	U
WR		19	17	A9	BUSREQ
-5V	BUSACK	20	18	A8	RES
WAIT		21	19		A7
+12V	BUSREQ	22	20		A6
V~	RESET	23	21		A5
M1		24	22		A4
RFSH		25	23		MEM ROM
A8		26			BUSACK
A10		27			A9
NC		28			A11

Emil Obermayr, Teichmüllerstraße 2, 3300 Braunschweig


ZX INTERFACE 1
Schaltplan
von Emil Obermayr
NET
Adressleitungen zur CPU
Datenleitungen zur CPU
XXOX,0XXX,000X,X000 ≙ CS=1
Adressleitungen
IC 4
CON 2
CON 1
Bauteil-seite
Löt-seite
microdrive
RS 232
ROM:
CE — A11 (Adressen)
Vpp — A12
R22
R23
Abschlußwiderstände am Schaltkontakt der NET-Buchsen

Und noch ein paar Konstante ...

n SPACES	gibt n-mal SPACE aus
HERE	Variable, deren Wert auf die Wörterbuchspitze zeigt
FORGET XXXX	löscht alle Wörter im Wörterbuch AB 'XXXX'
SMUDGE	invertiert das Smudge-Bit der letzten Definition
n1 n2 U<	Kleiner-als-Vergleich für vorzeichenlose Zahlen
n 0<	Test ob negativ
n 0=	Test ob Null

Noch was zum Thema Zahlen.
Jedes Element des Stacks belegt zwei Bytes. Dies gilt auch für ASCII-Codes, Flags und einfache Peeks ('C$'), welche mit nur mit dem High-Byte=0 abgelegt werden. Flags sind Null, wenn sie falsch sind, und 1 (oder auch -1, je nach 4th-System), wenn sie wahr sind. Zwischen vorzeichenlosen und vorzeichenbehafteten Zahlen besteht auf dem Stack kein Unterschied. Vorzeichenbehaftete Zahlen liegen im Bereich von -32768 bis +32767 und vorzeichenlose Zahlen von 0 bis 65535. Daher 'U' wie 'Unsigned' : U. , U< , etc.
Aufpassen:

```
32760 32770 U< .     => 1
32760 32770 < .      => 0   ( da 32770 als -32766 angenommen wird )
```

Nun wird's langsam Zeit, zu verzweigenden Programmbefehlen zu kommen.
IF-Beispiel in BASIC:

```
  IF A=2 THEN PRINT "OK" ELSE PRINT "AGAIN"
```

das gleiche in 4th:

```
  A $ 2 = IF ." OK" ELSE ." AGAIN" ENDIF
```

'IF' überprüft, ob ein True-Flag auf dem Stack liegt und führt in diesem Fall die Befehle hinter 'IF' aus, bis 'ELSE' oder 'ENDIF'. Wenn 'ELSE' verwendet wird, werden bei False die Befehle hinter 'ELSE' ausgeführt, bis 'ENDIF'.
Beispiele:

```
 ADR1 $ ADR2 $ U< IF ." KLEINER" ENDIF
 KEY DUP 65 = IF ." TASTE A" ELSE DUP 66 = IF ." TASTE B" ENDIF
 ENDIF DROP
 ( wartet auf Taste, bei 'A' wird 'TASTE A' ausgegeben, wenn nicht 'A', dann
 wird auf 'B' geprüft und wenn wahr, 'TASTE B' ausgegeben )
 MEM $ 0= IF ." ENDE" THEN
 ( 'THEN' entspricht 'ENDIF' )
```

Das Äquivalent zu einer FOR/NEXT-Schleife in BASIC oder einer FOR-Schleife in PASCAL/MODULA ist in 4th die DO/LOOP- bzw. DO/+LOOP-Schleife. Bei 'DO' werden Start- und Endwert auf dem Stack übergeben. Durch 'I' wird innerhalb der Schleife der aktuelle Schleifenwert auf den Stack gelegt. 'LOOP' zählt um eins hoch, vergleicht mit dem Endwert, und wenn dieser grösser ist, wird die Schleife wieder durchlaufen. Wenn andere Zählwerte statt +1 ('STEP 1') verwendet werden sollen ('STEP -1','STEP 2'), wird '+LOOP' verwendet, dem der Zählwert auf dem Stack übergeben wird.
Beispiele:

```
 3 1 DO I . LOOP      => 1 2
 5 1 DO I . 2 +LOOP => 1 3
 3 1 DO I 8 6 DO DUP . I . LOOP DROP LOOP   => 1 6 1 7 2 6 2 7
```

Und nun wird geübt:
Aufgabe 1:
Definiere eine Konstante 'ROM' mit dem Wert 5433 dezimal.
Lösung:

```
 DECIMAL 5433 CONSTANT ROM
```

Aufgabe 2:
Definiere ein Wort '.STR', das von einer auf dem Stack übergebenen Adresse an 24 Bytes aus dem Speicher als ASCII-Zeichen darstellt.

Lösung:

```
 : .STR DUP 24 + SWAP DO I C$ EMIT LOOP ;
```

Aufgabe 3:
Gebe ab der Adresse 5433 (s. Aufgabe 1) 24 Bytes als ASCII-Zeichen aus (s. Aufgabe 2). Abschluss mit Wagenrücklauf.
Lösung:

```
 ROM .STR CR
```

Aufgabe 4:
Neue Definitionen, Variablen, etc. erweitern das Wörterbuch. Sie werden an die Spitze des Wörterbuchs gesetzt, so daß sich der Zeiger auf die Spitze verändert. Wieviele Bytes belegt eine Variable 'TEST' ?
Lösung:

```
 HERE 1 VARIABLE TEST HERE SWAP - .
```

Aufgabe 5:
Definiere ein Wort, das auf Dualzahlen umschaltet, und ein Wort, das die aktuelle Zahlenbasis dezimal ausgibt, jedoch ohne sie zu ändern.
Lösung:

```
DECIMAL : DUAL 2 BASE ! ; : BASE? BASE $ DECIMAL DUP U. BASE ! ;
```

Fein geübt ! Und jetzt weiter mit der Theorie. Thema Schleifen. Die REPEAT/UNTIL-Schleife aus PASCAL/MODULA hat natürlich auch einen Verwandten in 4th. Hier nennt sie sich BEGIN/UNTIL-Schleife.
zum Bleistift:

```
 DECIMAL 49152 CONSTANT INPBUF
 : INPUT INPBUF CR BEGIN 1+ DUP KEY DUP DUP EMIT ROT C! 13 -
 UNTIL DUP INPBUF 1+ - INPBUF C! 0 SWAP C! ;
```

'BEGIN' markiert den Anfang der Schleife und 'UNTIL' das Ende (irgendwie logisch, oder?). 'UNTIL' erwartet ein Flag an der Spitze des Stacks (wir üben Englisch: Top of Stack). Ist das Flag falsch, so wird die Schleife wiederholt. Wenn das Flag wahr ist, wird die Programmausführung HINTER 'UNTIL' fortgesetzt. Im vorhergehenden Beispiel werden solange Zeichen von der Tastatur eingelesen und im Eingabepuffer 'INPBUF' abgelegt, bis die Enter-Taste gedrückt wird ('13 - UNTIL').
Anm.: '1+' und '2+' sind in den meisten 4th-Systemen schnelle Primitive, die um eins bzw. zwei inkrementieren ('1 +','2 +').
Aufgabe 6:
Beschreibe genau die Funktion der Wortdefinition 'INPUT' (s.o.).
Lösung:
'INPUT' gibt ein CR aus und liest dann solange Zeichen von der Tastatur, bis <enter> gedrückt wurde. Die Zeichen werden in dem durch die Konstante 'INPBUF' bestimmten Inputbuffer in Richtung grösser werdender Adressen abgelegt. Das erste Byte des Buffers enthält dann ein Zählbyte, das der Menge der eingegebenen Zeichen ohne <enter> entspricht, das zweite Byte ist das erste eingegebene Zeichen, etc. Als Endmarkierung wird nicht der ASCII-Code von <enter> (13), sondern 0 abgespeichert.

Jetzt möchte ich nocheinmal kurz auf den Interpreter-Compiler-Dualismus in 4th eingehen. 4th kennt zwei Modi: den Ausführungs- und den Kompilationsmodus. Der Ausführungsmodus ist der "Normale", in dem alle direkt ausgeführten Worte eingegeben werden. Der Kompilationsmodus ist z.B. innerhalb einer ':'-Definition aktiv, da alle Wörter nach einem ':' nicht direkt ausgeführt, sondern kompiliert werden, bis zum Abschluss durch ';'. Anders formuliert: Erst ';' schaltet wieder in den Ausführungsmodus zurück (oder auch einige Fehlermeldungen). Das heisst, daß man ohne Probleme eine Wortdefinition von vielleicht 300 Zeichen eingeben kann, auch wenn der entsprechende Editor nur 80, 64, oder auch nur 32 Zeichen auf einmal liest, weil sein Eingabepuffer nicht grösser ist. Man kann zum Beispiel am Ende jeder Zeile <enter> drücken, und dann in der nächsten Zeile einfach weitertippen, da nicht <enter>, sondern ';' die Definition beendet. Der einzige Unterschied zur Eingabe im Ausführungsmodus ist, daß dort nach jedem <enter> (und den eingegebenen Befehlen) ein 'OK' ausgegeben wird (oder eine Fehlermeldung). Die Variable 'STATE' enthält den entsprechenden Wert (0=Ausführungsmodus).

Jetzt freuen wir uns alle, daß wir auch lange Definitionen eingeben können, ... und ärgern uns, daß wir nach der Kompilation vielleicht gar nicht mehr wissen, wie die Definition aussah, oder wir wollen nur eine kleine Änderung vornehmen, und müssen dafür die komplette Definition neu eingeben. Aber (fast) jedes 4th-System hat ja noch einen Editor, der entweder zeilen-, bildschirm-, oder screen-orientiert ist (hier: Bildschirm <> Screen).Aber es gibt so viele verschiedene Editoren, wie es BASIC-Dialekte gibt oder Systemfehler beim Amiga, so daß es nicht möglich ist, universell darauf einzugehen. Da muß jeder selber die Anleitung für sein 4th zu Rate ziehen. Trotzdem möchte ich hier die Grundmerkmale eines "typischen" FIG-4th-Editor erklären.

FIG-4th arbeitet mit Screens, virtuellen Diskettenspeichern. Für ein 4th-System ist ein Screen ein Block einer Diskette oder Festplatte. Ob wirklich ein laufwerk oder eine Festplatte angeschlossen ist, ist je nach Implementierung der Disketten-Zugriffs-Worte egal. Die entsprechenden Worte können auch so definiert werden, daß sie eine RAM-Disk ansprechen. So funktioniert das auch bei allen mir bekannten 4th-Systemen für den Spectrum. Ein Screen besteht aus 8 oder auch 16 Zeilen zu je 32, 64 oder noch mehr Zeichen (je nach System).

Editor-Befehle
----

n LIST	macht n zum aktuellen Screen und listet ihn
n CLEAR	macht n zum aktuellen Screen und löscht ihn
SCR	Variable, die die Nummer des aktuellen Screens hält
n P XXXX	PUTtet den folgenden Text XXXX in die n-te Zeile des aktuellen Screens
n D	löscht die n-te Zeile, schiebt auf
n E	löscht die n-te Zeile, füllt mit Spaces
n LOAD	n-ten Screen "von Disk laden", -> kompilieren
-->	zeigt am Screen-Ende an, daß der nächste Screen auch kompiliert werden soll
FLUSH	schreibt geänderte Screens aus Puffer auf Disk
SAVE-BUFFERS	wie FLUSH
WHERE	zeigt nach einem LOAD-Kompilations-Fehler den Ort

Mir fallen keine schönen Hausaufgaben mehr ein ...

HA3:
Schreibe einen Hex-Dump.
HA4:
Definiere ein Wort, das zur letzten Definition beliebig viele Bytes dazufügt, um z.B. einer Variable noch ein Byte anzuhängen, welches ein Flag o.ä. aufnehmen kann. Das Wort soll praktisch nur den Zeiger erhöhen.
HA5:
Definiere ein Wort, das eine auf dem Stack liegende Zahl hinter der letzten Definition abspeichert und den Zeiger erhöht.
HA6:
Denke Dir eine Möglichkeit aus, welches Format Fliesskommazahlen mit 4 Nachkommastellen haben könnten. Definiere ein Wort, welches zwei der in dem ausgedachten Format auf dem Stack liegenden Fliesskommazahlen mit 1 (oder 2) Nachkommastellen Genauigkeit multipliziert.

Viel Spaß bis zum nächstenmal !

Hallo Pieter, machst Du auch mit ? Ich habe bis jetzt nichts von Dir bekommen; werde Dir mal Unterlagen schicken.

Noch ein letztes: SUCHE IMMER NOCH SPECTRUM-SCHROTT !!!

Wer hat Interesse an einer Harddisk ? Melden !

....

Frank Meurer, Schulstr. 21, 5047 Wesseling, Tel. 02236/46966

Tips zum Pokes finden

Endlich habe ich auch einmal einen Beitrag zur 'R.U.' fertiggestellt. Erst einmal zur Person:

Name: Patrick Thiel
Alter: 15 Jahre
Adresse: Königsberger Straße 11 / 4796 Salzkotten
Hobbys: Spectrum 128K, Atari ST 1040, Skateboardfahren

In diesem Beitrag möchte ich die Methode vorstellen, wie ich die Pokes für Robocop, Wec Le Mans etc. (Moonlight Madness, hi Hermann Mayr!) gefunden habe. Wenn es bessere Möglichkeiten gibt, Pokes zu finden, bitte ich euch User, sie zu veröffentlichen.
Voraussetzung für ein vernünftiges Ergebnis ist ein Multiface, ein schnellladender Datenträger (z.B. Microdrive) und ein wenig freie Zeit. Als erstes fertigt man mit dem Multiface eine Kopie des zu lösenden Spiels an und versieht sie mit folgendem Loader:

```
1 REM 48 K-Loader by the Cat
2 CLEAR VAL "24791": LOAD ""CODE: LOAD ""CODE: RANDOMIZE USR VAL
  "24830": LOAD ""CODE VAL "16464"
3 CLEAR VAL "28000"
```

oder für 128 K-Spiele (2. Basic-Loader verändern und von ihm angefangen laden):

```
1 REM 128 K-Loader by the Cat
2 CLEAR VAL "25047": LOAD ""CODE: LOAD ""CODE: RANDOMIZE USR VAL
  "25086": LOAD ""CODE VAL "16470"
3 CLEAR VAL "28000"
```

Nun kann das Spiel geladen und folgendes Programm dazugefügt werden:

```
 1 REM (C) 1990 THE CAT
10 BORDER 0: PAPER 0: INK 6: CLS: LET z=0
20 INPUT "STARTADRESSE (>31000): ";u
30 INPUT "ENDADRESSE (<65535): ";v
40 FOR x=u TO v: FOR y=0 TO 0: IF PEEK x<>205 OR PEEK (x+2)<90 THEN NEXT
   x
50 IF PEEK x=205 THEN FOR y=30000 TO 30000+2*z STEP 2: IF PEEK y=PEEK (x
   +1) AND PEEK (y+1)=PEEK (x+2) THEN NEXT x
60 NEXT y: POKE 30000+2*z,PEEK (x+1): POKE 30000+2*z+1,PEEK (x+2): PRINT
   AT 0,0;z;"  ";x: LET z=z+1: NEXT x
70 PRINT: PRINT FLASH 1;"Adresse, Inhalt": PRINT: INK 7: FOR x=30000 TO
   30000+2*(z-1) STEP 2: PRINT PEEK x+256*PEEK (x+1);",";PEEK (PEEK x+
   256*PEEK (x+1));"     "; INK 3;"POKE NUMMER ";(x-30000)/2: BEEP .006,
   10: BEEP .006,6: NEXT x
```

Wer über einen Drucker verfügt, sollte die Prints in Zeile 70 in Lprints umwandeln.
Zur Funktion: Das Programm durchsucht den Speicher nach einem CALL NN-Befehl. Es sortiert die gefundenen Adressen ab 30000 und gibt so ab Zeile 70 jede gefundene Adresse nur einmal aus. ROM-Adressen werden nicht beachtet. Während der erste Teil des Programms läuft, kann man sich übrigens prima ein paar Artikel der 'RU' reinziehen, denn den Speicher in Basic zu durchforsten dauert eine Weile. Ist der erste Teil abgeschlossen und die Adressen werden aufgelistet, muß man sich die Adressen und deren Inhalte notieren (kann der Drucker übernehmen). Als Abschluß der Poke-Suche muß man während des Spiels einzeln die Pokes mit dem Wert 201 eingeben und sehen, wie sich das auf den Spielablauf auswirkt. Kommen unerwünschte Folgen zustande (z.B. Sprites unsichtbar), kann man den ursprünglichen Wert wieder poken und sich die nächste Adresse vorknüpfen (ausgenommen natürlich nach einem Absturz, welcher leider öfter vorkommt, deshalb schnelladender Datenträger!). Zum Schluß wären die Pokes dann noch nach

Köln (zum WOMO-Team) zu schicken und man hat einen Beitrag geleistet, die 'RU' mit zu erstellen.
Falls es Probleme gibt, hier nochmal meine Telefonnummer: 05258/5197.

Auf diese Art und Weise habe ich für Herman Mayr folgende Pokes für das Spiel "Moonlight Madness" herausgefunden:
25780,x (x=Anzahl der Leben), 57931,0 + 57932,0 + 57933,0 für gefahrlose Sprünge und 62542,201 (durch die Gegner gehen).

So, das wäre geschafft. Hier noch ein kleiner Tip, wie man Programme saven oder Cartdridges formatieren kann. Erstmal die Tabelle:

```
CHR$(6)  = verschiebt um 16 Zeichen nach rechts
CHR$(16) = Ink (+ CHR$(0-9))
CHR$(17) = Paper (+ CHR$(0-9))
CHR$(18) = Flash (+ CHR$(0-1))
CHR$(19) = Bright (+ CHR$(0-1))
CHR$(20) = Inverse (+ CHR$(0-1))
CHR$(21) = Over (+ CHR$(0-1))
CHR$(22) = Art (+ CHR$(x)+CHR$(y))
CHR$(23) = Tab (+ CHR$(0-255))
```

So könnte dann das Absaven eines Basicprogramms aussehen:

```
SAVE CHR$(22)+CHR$(1)+CHR$(0)+"CAT"
```

(Beim Laden wird der Name an die Koordinate 1,0 gesetzt!).

Zum Schluß noch ein Appell an Dieter Hucke: Fordere doch noch einmal zu einem MC-Kurs auf. Wir haben 43 neue User und als ich damals den Aufruf las, hätte ich dir geschrieben, aber als "von Natur aus (schreib-) faul" habe ich gedacht: "Das erledigt sich schon.". Ich fände es wirklich super, demnächst meine Programme im MC zu veröffentlichen (bis heute hat mir U.S.C.H.I. geholfen!). Auf bald,

---

Hallo Leute !

Da ich jetzt endlich dazu gekommen bin, dem SPC beizutreten, möchte ich mich auch einmal kurz vorstellen.

Name:	Hanno Foest, 22 Jahre, Student (Physik)
System:	Spectrum+, 3 * Spectrum 48K, Gummitastatur hab' ich auch noch
Peripherie:	IF 1, einige Microdrives (gibt's noch neue Cartridges in England oder sonstwo?), RGB-Interface (Funkschau 6/87), Kempston Joystick, Kempston-Centronics-Interface E, ZON-X Soundmodul
Drucker:	Commodore (würg!) 8028 Typenraddrucker, leider z. Zt. nicht am Studienort
Sonstige Computer:	ZX 81, Jupiter ACE (suche Handbuch!), NDR-Klein-Computer, einiges in Richtung IBM XT/AT-Kompatible (zusammengeflickter Schrott; sowas bekommt man ja in der letzten Zeit nachgeschmissen).
Hobbys:	Elektronik- und Computerbasteln, Flens, Science Fiction,FGTH, Sammeln von Sinclair-Geräten, Sammeln von älteren Computern und deren Periperie, Fahrradbasteln,
Pseudonym in CF:	The Fraggle

Bei meinen Hardwarebasteleien ist als nächstes der ULA-Nachbau (die Schaltung aus der DDR von Markus Haupt) dran. Ein Floppy-Interface suche ich noch: wer hat Opus- und Disciple-Schaltpläne? Außerdem suche ich Spectrum 128K Unterlagen (Schaltpläne etc., wie geht das Bankswitching?) sowie ein kommentiertes IF 1-ROM Disassembly. Ein defekter Spectrum 128K wäre auch nicht schlecht.
Ich hab zwar immer ziemlich viel zu tun, hoffe aber trotzdem, mal was für den Rainbow User schreiben zu können.
Bis demnächst, Hanno.

Hanno Foest, Querumer Str. 41, 3300 Braunschweig

Gamma-NMI-Files-Loader

Heute habe ich eine Routine, mit der es möglich ist, NMI-Snapshot-Files eines Gamma-Interfaces (ß-Disk kompatibel) auch <u>ohne angeschlossenes Gamma-Interface</u> zu laden. Somit kann man die Files auch auf Tape saven oder über RS 232 in andere Speccies bringen usw.
Das Programm I wird inclusive REM-Zeile (65 Zeichen) eingegeben und laufen gelassen. Dann werden alle Zeilen bis auf die REM-Zeile (die jetzt Zeile 0 ist) entfernt.
<u>Dazu</u> dann die Zeilen 10-40 (+ 15 für Tape) des Listings II eingeben und absaven.
Nun wird dieser LOADER als Ersatz des normalen LOADERS benutzt (Einfach "name2", "name1" dementsprechend abändern und LOADER absaven.

PROGRAMM I

```
   1 REM (hier 65 Zeichen eingeben)
  20 LET basic=PEEK 23635+256*PEEK 23636
  25 LET mcode=basic+6
  30 PRINT basic,mcode
  40 RESTORE: FOR a=mcode TO mcode+62
  50 READ b: POKE a,b: NEXT a
  60 POKE (basic+1),0: POKE mcode-1,13
1000 DATA 197,225,17,116,64,1,63,0,237,176
1010 DATA 195,129,64,243,33,0,65,17,0,91
1020 DATA 1,0,7,237,176,49,21,64,225,209
1030 DATA 193,217,241,8,253,225,221,225,209,193
1040 DATA 241,226,161,64,251,254,63,237,71,40
1050 DATA 2,237,94,241,237,79,225,241,237,123
1060 DATA 45,64,201
```

PROGRAMM II

```
   0 REM (Zeile aus Programm I)
  10 CLEAR 25087
  20 RANDOMIZE USR 15363: REM : LOAD "name2"CODE
  30 RANDOMIZE USR 15363: REM : LOAD "name1"CODE
  40 RANDOMIZE USR ((PEEK 23635+256*PEEK 23636)+6)
```

Für Cassettenprogramme muß in Programm II noch folgende Zeile eingefügt werden, welche den Ausdruck des Programmnamens auf dem Bildschirm verhindert:

```
  15 POKE (PEEK 23631+256*PEEK 23632+5),111
```

Und hier noch das Disassembly der "RUNNER-ROUTINE" aus dem Gamma-ROM (Zeile 90-370):

```
010 *D+
020   ORG 16500
030   PUSH BC
040   POP  HL
050   LD   DE,16500
060   LD   BC,63
070   LDIR
080   JP   16513
090   DI
100   LD   HL,16640
110   LD   DE,23296
120   LD   BC,1792
130   LDIR
140   LD   SP,16405
150   POP  HL
160   POP  DE
170   POP  BC
180   EXX
190   POP  AF
200   EX   AF,AF'
210   POP  IY
220   POP  IX
230   POP  DE
240   POP  BC
250   POP  AF
260   JP   PO, OVER1
270 EI
280 OVER1   CP   #3F
290         LD   I,A
300         JR   Z, OVER2
310         IM   2
320 OVER2   POP  AF
330         LD   R,A
340         POP  HL
350         POP  AF
360         LD   SP,(16429)
370         RET
380         NOP
```

Bernhard Lutz, Hammerstraße 35, 6729 Bellheim, Tel. 07272/1444

Das Opus-Discovery Laufwerk Teil 2

Im letzten Teil meiner kleinen Serie haben wir uns mit dem Aufbau einer formatierten Opus-Diskette beschäftigt (Übrigens sind bei fast allen Diskettensystemen die Disketten ähnlich formatiert; bei einer Opus 180K könnten einige Dinge, die ich hier beschreibe ein wenig anders sein).

Heute möchte ich den CAT-Befehl -und das was er bewirkt- etwas näher unter die Lupe nehmen.

Der gesammte Katalog (und einige Informationen, die bei "CAT 1" nicht ausgegeben werden, auf die ich aber später noch eingehen möchte) steht am Anfang der Diskette. Dabei sind einige Sektoren, die ganz am Anfang der Diskette stehen für den Katalog reserviert. Man kann diesen Katalog auch ohne CAT 1 einzugeben herauslesen. Diese Methode hat den Vorteil, daß man die Zusatzinformationen, von denen eben die Rede war auch lesen kann und den Katalog nicht direkt auf dem Bildschirm hat, sondern auch z.B. in einen String einlesen kann. Mit der Befehlsfolge OPEN #4;" CAT ";1 RND 16 kann man das Katalogfile öffnen (Vorsicht im 128K Modus! Da man dort alle Befehle Buchstabenweise eingibt, liest der Computer " CAT " nicht als den Befehl CAT (=CHR$ 207) sondern als einzelne Buchstaben c,a,t. Da dies der Computer nicht versteht, sollte man diese Zeilen im 48K Modus eingeben).

Wenn man nun das Katalogfile geöffnet hat kann man beliebig in diesem File lesen und auch schreiben (Vorsicht! Wer an die falsche Stelle im Katalogfile etwas falsches hinschreibt kann eine böse Uberraschung erleben; z.B. einen I/O Error bei CAT 1, dann ist fast nichts mehr zu retten...) Mit POINT #4;<Zahl> zeigt man dann auf ein Programm, das im Katalog steht. Wenn man nun alle Informationen über das Programm, auf das der Zeiger momentan zeigt erhalten möchte, dann kann man mit:

```
DIM a$(16): FOR x=1 TO 16: LET a$(x)=INKEY$ #4: NEXT x
```

die Informationen in a$ schreiben.
Hierbei geben die letzten 10 Bytes von a$, also a$(7 TO 16) den Namen des Programms an, von dem man gerade die Infos gelesen hat. Die ersten sechs Bytes geben die Zusatzinfos an, von denen ich eben gesprochen habe. Dabei ergeben immer zwei Bytes Zusammengefasst eine Nummer (also Byte 1 und 2, Byte 3 und 4, Byte 5 und 6), die mit LET a=CODE a$(x)+CODE a$(x+1)*256 berechnet werden kann (x ist hier 1,3 oder 5).
Die Zahlen, die man hierbei bekommt haben folgende Bedeutung:

- Byte 3 und 4 geben die Nummer des ersten Sektors an, auf dem das Programm abgespeichert ist.Auf diesem und den folgenden Sektoren steht dann das Programm. Natürlich muß der Computer nun wissen, wieviele Sektoren für dieses Programm reserviert sind.
- Dafür sind Bytes 5 und 6 zuständig. Sie geben an welches der letzte Block ist auf dem das Programm steht (einschliesslich). Nun weiß der Computer genau, ab welchem Sektor (Bytes 3 und 4) bis zu welchem Sektor (Bytes 5 und 6) er das Programm zu suchen hat.
- Der letzte Sektor, der von dem Programm beansprucht wird hat nun eine besondere Eigenschaft. Leider sind nicht alle Programme genau so lang, daß der letzte Sektor bis zum letzten Byte mit dem Programm vollgeschrieben wird. Damit der Comuter nun erkennen kann wieviele Bytes am Ende noch frei sind, gibt es Bytes 1 und 2. Diese Zahl gibt an wieviele Bytes im letzten Block belegt sind. Jedoch zählen hierfür nur die ersten 12 Bits dieses Doppelbytes. Die letzten vier werden vom System benötigt (mehr kann ich leider über diese vier Bits auch nicht sagen, da ich es selbst nicht genau weiß)

Der erste Eintrag in dem Katalogfile (POINT #4;1) spielt eine besondere Rolle. Es handelt sich hierbei um kein Programm, das auf der Diskette abgespeichert

wurde, sondern um den Namen der Diskette, der bei CAT 1 ganz oben erscheint. Auch hier ist das Ergebnis beim Einlesen 16 Bytes lang. Bytes 7 bis 16 geben hier den Namen der Diskette an. Die Bedeutung von Bytes 1 bis 6 ist mir leider unbekannt.(es ist auch möglich, daß sie gar keine Bedeutung haben).
Wir wissen jetzt wie man den Namen der Diskette und die Informationen über ein Programm einliest. Wie erkennt man aber das letzte File, damit man z.B. bei einer Schleife nicht endlos nach weiteren Files sucht, obwohl man schon alle Programmnamen eingelesen hat? Dieses Problem ist mit einem sogenannten End Marker gelöst worden. Hinter dem letzten Eintrag befindet sich noch ein weiterer Eintrag, bei dem Bytes 5 und 6 die Zahl 65535 (oder HEX FFFF) ergeben. Dieses Ergebnis kommt normalerweise nicht vor, so daß man so erkennen kann, daß das Ende erreicht wurde. Bytes 3 und 4 geben die Anzahl der restlichen Sektoren an die noch frei sind. Bytes 1,2,7 bis 16 haben bei dem End Marker keine Bedeutung.
Damit wären wir am Ende des heutigen Teils angelangt. Das nächste mal wird wohl der letzte Teil dieser Serie sein, in dem ich darauf eingehen werde, wie man weitere Informationen über ein Programm erhalten kann. So, das war's von mir für heute, bis zum nächsten Mal,

Rüdiger Döring, Meisenstraße 10, 5467 Vettelschoß, Tel.: 02645/3060

Anzeigen

NEURONALE NETZE
sind Gegenstand internationaler Konferenzen. Ich kann nicht glauben, daß diese Entwicklung ganz ohne SPECCI-Fans vonstatten geht.
Im PM-Magazin 4/90 stand ein lesenswerter Aufsatz zu Neuronalen Netzen. Wer kann mir in dieser Richtung weiterhelfen ?

FUZZY LOGIC
Vor 25 Jahren hafte Lofti A. Zadeh als Professor der University of California in Berkeley die FUZZY-LOGIC-Theorie entwickelt. Die FUZZY-LOGIC verarbeitet unscharfe Informationen. Amerikanische und japanische Elektronikfirmen arbeiten an FUZZY-LOGIC-Steuerwerken, die die Verarbeitung von unpräzisen Angaben, wie "meistens", "selten", "viele" und "wenige" erlaubt. SONY hat jüngst einen Taschenrechner herausgebracht, der nach dieser Logik funktioniert. Wer kann mir bei diesem Thema weiterhelfen ?

Peter Keller-Bentzen, Karlstal 11, 7452 Haigerloch, Tel. 07474/8586

Habe ein 40-Spur doppelseitiges Laufwerk komplett anschlußfähig an Beta-Disk Interface 5 1/4 Zoll für 150,- DM inkl. Porto.
Basic-Handbuch, eigentlich noch gut erhalten für 25,- DM inkl. Porto.

Hermann Mayr, Grafstraße 2/4, 8025 Unterhaching, Tel. 089/618924

Suche für mein "Fußall WM 1990"-Programm die Nationalhymnen aller Teilnehmerländer. Kann mir da jemand in Form von 48er (oder 128er Tönen), bzw. Noten oder einer Kassettenaufnahme helfen ?
Hat überhaupt schon jemand von Euch ein Programm mit Hymnen geschrieben ?
Ferner interessieren mich nach wie vor Eure 128er Sounds, das Echo auf die bisherigen Anfragen war eher bescheiden.
Oder seid ihr gar alle so furchtbar unmusikalisch ?

Wolfgang Haller, Ernastraße 33, 5000 Köln 80, Tel. 0221/685946

Suche Beta-Disk mit oder ohne Laufwerk !

Uwe Riemer, Winterstraße 2, Dresden, 8030, DDR

Hallo Fraggle ! Vielen Dank für die Milka. Mo (von Wo).
